Improving the Robustness of Reinforcement Learning Policies With $\mathcal{L}_1$ Adaptive Control

Yikun Cheng, *Student Member, IEEE*, Pan Zhao, *Member, IEEE*, Fanxin Wang, Daniel J. Block, and Naira Hovakimyan, *Fellow, IEEE*

***Abstract*—A reinforcement learning (RL) control policy could fail in a new/perturbed environment that is different from the training environment, due to the presence of dynamic variations. For controlling systems with continuous state and action spaces, we propose an add-on approach to robustifying a pre-trained RL policy by augmenting it with an $\mathcal{L}_1$ adaptive controller ($\mathcal{L}_1$AC). Leveraging the capability of an $\mathcal{L}_1$AC for fast estimation and active compensation of dynamic variations, the proposed approach can improve the robustness of an RL policy which is trained either in a simulator or in the real world without consideration of a broad class of dynamic variations. Numerical and real-world experiments empirically demonstrate the efficacy of the proposed approach in robustifying RL policies trained using both model-free and model-based methods.**

***Index Terms*—Reinforcement learning, machine learning for robot control, robust/adaptive control, robot safety.**

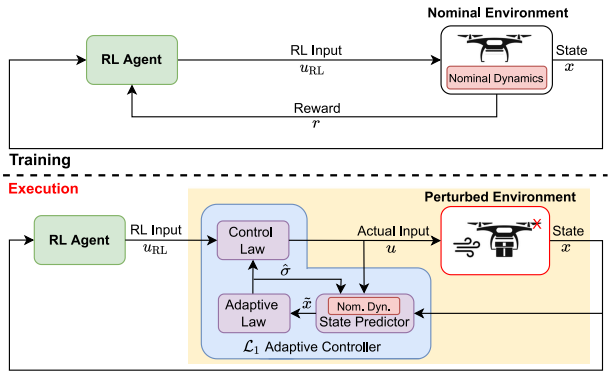

Fig. 1. Proposed approach to policy robustness improvement based on $\mathcal{L}_1$ adaptive augmentation.

I. Introduction

REINFORCEMENT learning (RL) is a promising way to solve sequential decision-making problems [1]. In the recent years, RL has shown impressive or superhuman performance in control of complex robotic systems [2], [3]. An RL policy is often trained in a simulator and deployed in the real world. However, the discrepancy between the simulated and the real environment, known as the sim-to-real (S2R) gap, often causes the RL policy to fail in the real world. An RL policy may also be directly trained in a real-world environment; however, the environment perturbation resulting from parameter variations, actuator failures and external disturbances can still cause the well-trained policy to fail. Take a delivery drone for example (Fig. 1). We could train an RL policy to control the drone in a nominal environment (e.g., nominal load, mild wind disturbances, healthy propellers, etc.); however, this policy could fail and lead to a crash when the drone operates in a new environment (e.g., heavier loads, stronger wind disturbances, loss of propeller efficiency, etc.). To a certain extent, the S2R gap issue can be considered as a special case of environment perturbation by treating the simulated and real environments as the old/nominal and new/perturbed environments, respectively.

Manuscript received December 5, 2021; accepted April 11, 2022. Date of publication April 21, 2022; date of current version May 24, 2022. This work is supported in part by AFOSR, and in part by NSF under the RI grant 2133656, NRI grant 1830639 and AI Institute Planning grant 2020289. This letter was recommended for publication by Associate Editor M. Khoramshahi and Editor J. Kober upon evaluation of the reviewers' comments. *(Yikun Cheng and Pan Zhao contributed equally to this work.) (Corresponding author: Pan Zhao.)*

Yikun Cheng, Pan Zhao, Fanxin Wang, and Naira Hovakimyan are with the Mechanical Science and Engineering Department, University of Illinois at Urbana-Champaign, IL 61801 USA (e-mail: yikun2@illinois.edu; panzhao2@illinois.edu; fanxinw2@illinois.edu; nhovakim@illinois.edu).

Daniel J. Block is with the Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign, IL 61801 USA (e-mail: d-block@illinois.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3169309, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3169309

A. Related Work

Robust/adversarial training: Domain/dynamics randomization was proposed to close the sim-to-real (S2R) gap [4]–[6] when transferring a policy from a simulator to the real world. Robust adversarial training addresses the S2R gap and environment perturbations by formulating a two-player zero-sum game between the agent and the disturbance [7]. A similar idea was explored in [8], where Wasserstein distance was used to characterize the set of dynamics for which a robust policy was searched via solving a min-max problem. Though fairly general and applicable to a broad class of systems, these methods often involve tedious modifications to the training environment or the dynamics, which can only happen in a simulator. More importantly, the resulting *fixed* policies could overfit to the worst-case scenarios, and thus lead to conservative or degraded performance in other cases [9]. This issue is well studied in control community; more specifically, robust control [10] that aims to provide performance guarantee for the worst-case scenario, often leads to conservative nominal performance.

Post-training augmentation: Kim *et al.* [11] proposed to use a disturbance observer (DOB) to improve the robustness of

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

an RL policy, in which the mismatch between the simulated training environment and the testing environment is estimated as a disturbance and compensated for. A similar idea was pursued in [12], which used a model reference adaptive control (MRAC) scheme to estimate and compensate for parametric uncertainties. Our objectives are similar to the ones in [11] and [12], but our approach and end results are different, as we address a broader class of dynamic uncertainties (e.g., unknown input gain that cannot be handled by [11], and time-dependent disturbances that cannot be handled by [12]), and we leverage the $\mathcal{L}_1$ adaptive control architecture that is capable of providing guaranteed transient (instead of just asymptotic) performance [13]. Additionally, we validate our approach on real hardware, as opposed to merely in numerical simulations in [11], [12]. We note that $\mathcal{L}_1$ adaptive control has been combined with model predictive control (MPC) with application to quadrotors [14], and it has been used for safe learning and motion planning applicable to a broad class of nonlinear systems [15]–[17]. To put things into perspective, this paper is focused on applying the $\mathcal{L}_1$ adaptive control architecture to robustify an RL policy. In terms of technical details, this paper considers more general scenarios, e.g., unmatched disturbances and unknown input gain, which were not considered in [16], [17].

Learning to adapt: Meta-RL has recently been proposed to achieve fast adaptation of a pre-trained policy in the presence of dynamic variations [18]–[22]. Despite impressive performance mainly in terms of fast adaptation demonstrated by these methods, the intermediate policies learned during the adaptation phase will most likely still fail. This is because a certain amount of information-rich data needs to be collected in order to *learn* a good model and/or policy. On the other hand, rooted in the theory of adaptive control and disturbance estimation, [13], [23], [24], our proposed method can *quickly estimate* the discrepancy between a nominal model and the actual dynamics, and *actively compensate* for it in a timely manner. We envision that our proposed method can be combined with these methods to achieve robust and fast adaptation.

B. Statement of Contributions

For controlling systems with continuous state and action spaces, we propose an *add-on* approach to robustifying an RL policy, which can be trained in standard ways without consideration of a broad class of potential dynamic variations. The essence of the proposed approach lies in augmenting it with an $\mathcal{L}_1$ adaptive control ($\mathcal{L}_1$AC) scheme [13] that quickly estimates and compensates for the uncertainties so that *the dynamics of the system in the perturbed environment are close to that in the nominal environment*, in which the RL policy is trained and thus expected to function well. The idea is illustrated in Fig. 1. Different from most of existing robust RL methods using domain randomization or robust/adversarial training [4]–[8], the proposed approach can be used to robustify an RL policy, which is trained either in a simulator or in the real world, using both model-free and model-based methods, without consideration of a broad class of uncertainties in the training. We empirically validate the approach on both numerical examples and real hardware.

II. Problem Setting

We assume that we have access to the system dynamics in the *nominal* environment, either simulated or in the real world, and they are described by a nonlinear control-affine model:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \triangleq F_{\text{nom}}(x(t), u(t)), \quad (1)$$

where $x(t) \in \mathcal{X} \subset \mathbb{R}^n$ and $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ are the state and input vectors, respectively, $\mathcal{X}$ and $\mathcal{U}$ are compact sets, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^m \to \mathbb{R}^{n\times m}$ are known and locally Lipschitz-continuous functions. Moreover, $g(x)$ has full column rank for any $x \in \mathcal{X}$.

Remark 1: Control-affine models are commonly used for control design and can represent a broad class of mechanical and robotic systems. In addition, a control non-affine model can be converted into a control-affine model by introducing extra state variables (see e.g., [25]). Therefore, the control-affine assumption is not very restrictive.

The nominal model (1) can be from physics-based modeling, data-driven modeling or a combination of both. Methods exist for maintaining the control affine structure in data-driven modeling (see e.g., [26]).

Assumption 1: We have access to a *nominal* control policy, $\pi_o(x)$, which is trained using the *nominal* dynamics (1) and thus functions well under such dynamics. Moreover, $\pi_0(x)$ is Lipschitz continuous in $\mathcal{X}$ with a Lipschitz constant l_π.

The policy $\pi_o(x)$ can be trained either in a simulator or in the real world in the standard (i.e., non-robust) way, using either model-based and model-free methods. The Lipschitz continuity assumption is needed to derive an error bound for estimating the disturbances in Section III-D. The nominal policy π_0 could fail in the perturbed environment due to the dynamic variations. We, therefore, propose a method to improve the robustness of this nominal policy in the presence of such dynamic variations, by leveraging $\mathcal{L}_1$AC [13]. To achieve this, we further assume that the dynamics of the agent in the *perturbed* environment can be represented by

$$\dot{x} = f(x) + g(x)\Lambda u + d(t, x), \quad (2)$$

where Λ is an unknown input gain matrix, which satisfies Assumption 2, $d(t, x)$ is an unknown function that can capture parameter perturbations, unmodeled dynamics and external disturbances. It is obvious that the perturbed dynamics (2) can be equivalently written as

$$\dot{x} = F_{\text{nom}}(x, u) + \sigma(t, x, u), \quad (3)$$

where

$$\sigma(t, x, u) \triangleq g(x)(\Lambda - I)u(t) + d(t, x). \quad (4)$$

Remark 2: Uncertain input gain is very common in real-world systems. For instance, actuator failures, and variations in mass or inertia for force- or torque-controlled robotic systems, normally induce such input gain uncertainty. For a single-input system, $\Lambda = 0.6$ indicates a 40% loss of the control effectiveness. Our representation of such uncertainty in (2) is broad enough to capture a large class of scenarios, while still allowing for effective compensation of such input gain uncertainty using $\mathcal{L}_1$AC (detailed in Section III).

To provide a rigorous treatment, we make the following assumptions on the perturbed dynamics (2).

Assumption 2: The matrix Λ in (2) is an unknown strictly row-diagonally dominant matrix with $\text{sgn}(\Lambda_{ii})$ known. Furthermore, there exists a compact convex set $\not\geq$ such that $\Lambda \in \not\geq$.

Remark 3: The first statement in Assumption 2 indicates that Λ is always non-singular with known sign for the diagonal elements, and is often needed in applying adaptive control methods to mitigate the effect of uncertain input gain (see [23, Sections 6 and 7]). Without loss of generality, we further assume that $\not\geq$ in Assumption 2 contains the m by m identity matrix, I.

The problem we are tackling can be stated as follows. **Problem Statement**: Given an RL policy $\pi_o(x)$ well trained in a nominal environment with the nominal dynamics (1), assuming the dynamics in the perturbed environment are represented by (2) satisfying Assumption 2, develop an augmentation-based solution to improve the robustness of the policy $\pi_o(x)$ in the perturbed environment.

III. $\mathcal{L}_1$ Adaptive Augmentation for RL Policy Robustification

A. Overview of the Proposed Approach

The idea of our proposed approach is depicted in Fig. 1. With our approach, the training phase is standard: the nominal policy can be trained using almost any RL methods (both model-free and model-based) in a nominal environment. After getting a nominal policy that functions well in the nominal environment, for *policy execution*, an $\mathcal{L}_1$ controller is designed to augment and work together with the nominal policy. The $\mathcal{L}_1$ controller uses the dynamics of the nominal environment (1) as an internal nominal model, estimates the discrepancy between the nominal model and the actual dynamics and compensates for this discrepancy so that *the actual dynamics with the $\mathcal{L}_1$ controller* (illustrated by the shaded area of Fig. 1) *are close to the nominal dynamics*. Since the RL policy is well trained using the nominal dynamics, it is expected to function well in the presence of the dynamic variations *and* the $\mathcal{L}_1$ augmentation.

B. RL Training for the Nominal Policy

As mentioned before, the policy can be trained in the standard way, using almost any RL method including both model-free and model-based one. The only requirement is that one has access to the nominal dynamics of the training environment in the form of (1).

As an illustration of the idea, for the experiments in Section IV, we choose PILCO [27], a model-based policy search method using Gaussian processes, soft actor-critic [28], a state-of-the-art model-free deep RL method, and a trajectory optimization method based on differential dynamic programming (DDP) [29] to obtain the nominal policy.

C. $\mathcal{L}_1$ Adaptive Augmentation for Policy Robustification

In this section, we explain how an $\mathcal{L}_1$AC scheme can be designed to augment and robustify a nominal RL policy. An $\mathcal{L}_1$ controller mainly consists of three components: a state predictor, an adaptive law, and a low-pass filtered control law. The state predictor is used to predict the system's state evolution, and the prediction error is subsequently used in the adaptive law to update the disturbance estimates. The control law aims to compensate for the estimated disturbance. For the perturbed system (2) with the nominal dynamics (1), the **state predictor** is given by

$$\dot{\hat{x}}(t) = F_{\text{nom}}(x,u) + \hat{\sigma}(t) - a\tilde{x}(t), \tag{5}$$

where $\tilde{x}(t) \triangleq \hat{x}(t) - x(t)$ is the prediction error, a is a positive constant, $\hat{\sigma}(t)$ is the estimation of the lumped disturbance, $\sigma(t,x,u)$, at time t. Following the piecewise-constant (PWC) **adaptive law** (which connects with the CPU sampling time) [13, Section 3.3], the disturbance estimates are updated as

$$\begin{aligned} \hat{\sigma}(t) &= \hat{\sigma}(iT), \quad t \in [iT, (i+1)T), \\ \hat{\sigma}(iT) &= -\frac{a}{e^{aT}-1}\tilde{x}(iT), \end{aligned} \tag{6}$$

for $i = 0, 1, \cdots$, where T is the estimation sampling time. With $\hat{\sigma}(t)$, we further compute

$$\begin{bmatrix} \hat{\sigma}_m(t) \\ \hat{\sigma}_{um}(t) \end{bmatrix} = \left[g(x)\ g^{\perp}(x)\right]^{-1} \hat{\sigma}(t), \tag{7}$$

where $\hat{\sigma}_m(t)$ and $\hat{\sigma}_{um}(t)$ are the *matched* and *unmatched* disturbance estimates, respectively, $g^{\perp}(x) \in \mathbb{R}^{n-m}$ satisfies $g(x)^{\top} g^{\perp}(x) = 0$, and $\text{rank}([g(x)\ g^{\perp}(x)]) = n$ for any $x \in \mathcal{X}$. From (3) and (5), we see that the total or lumped disturbance $\sigma(t,x,u)$, is estimated by $\hat{\sigma}(t) \triangleq g(x)\hat{\sigma}_m(t) + g^{\perp}(x)\hat{\sigma}_u m(t)$. The **control law** is given by

$$\begin{aligned} u(t) &= u_{\text{RL}}(t) + u_{\mathcal{L}_1}(t), \\ u_{\mathcal{L}_1}(s) &= -C(s)\mathfrak{L}[\hat{\sigma}_m(t)], \end{aligned} \tag{8}$$

where $u_{\text{RL}}(t) = \pi_0(x(t))$ is the control command from the nominal RL policy, $u_{\mathcal{L}_1}(s)$ is the Laplace transform of the $\mathcal{L}_1$ control command $u_{\mathcal{L}_1}(t)$, $\mathfrak{L}[\cdot]$ denotes the Laplace transform, and $C(s) \triangleq K(sI+K)^{-1}$ is an m by m transfer matrix consisting of low-pass filters with $K \in \mathbb{R}^{m\times m}$.

Remark 4: As it can be seen from (5), (6), (8) in an $\mathcal{L}_1$AC scheme with a PWC adaptive law [13, section 3.3], all the dynamic uncertainties (such as parametric uncertainties, unmodeled dynamics and external disturbances) are *lumped* together and estimated as a total disturbance. This is different from most adaptive control schemes [23], which rely on a parameterization of the uncertainty to design adaptive laws for updating parameter estimates and usually consider only stationary uncertainties that do not directly depend on time.

Details on deriving the estimation and control laws can be found in [30], [31]. The working principle of the $\mathcal{L}_1$ controller can be summarized as follows: the state predictor (5) and the adaptive law (6) can accurately estimate the lumped disturbances, $\hat{\sigma}_m(t)$ and $\hat{\sigma}_{um}(t)$. In fact, under certain conditions, a bound on the estimation error, $\hat{\sigma}(t) - d(t,x)$, can be derived and is included in [32]. Additionally, the control law (8) mitigates the effect of disturbances by cancelling those within the bandwidth of the low-pass filter. Note that unmatched disturbances (also known as mismatched disturbances in the disturbance-observer based control literature [24]) cannot be directly canceled by control signals and are more challenging to deal with.

TABLE I
COMPARISON WITH EXISTING APPROACHES TO IMPROVING THE ROBUSTNESS OF RL POLICIES

	Robust/Adversarial Training [4]–[8]	**Post-Training Augmentation**		
		MRAC [12]	**DOB** [11]	**$\mathcal{L}_1$AC** (ours)
Complexity of training	High	Low		
Training environment	Simulated	Simulated & Real-world		
Restrictions on structure of dynamics	Low	High (control-affine & continuous)		
Control inputs	Multiple	Single	Single	Multiple
Restrictions on uncertainties	Low	High (matched parametric uncertainties)	High (matched disturbances)	Medium (matched uncertainties and disturbances)
Control policy after training	Fixed	Adapted Online		
Validation	Sims & Experiments	Sims	Sims	Sims & Experiments

Remark 5: In designing the $\mathcal{L}_1$ controller consisting of (5), (6), and (8), we assume that the states are measured without noise. In practice, as long as the estimation sampling time is not too small and the filter bandwidth is not too large, moderate measurement noise that always exists in real-world systems usually does not cause big issues, as demonstrated by the hardware experiments in Section IV-C.

Remark 6: Variants of the proposed $\mathcal{L}_1$AC law (5), (6), and (8) have been used to augment other baseline controllers (e.g., PID, linear quadratic regulator, MPC), as demonstrated in numerous applications and flight tests, [13].

D. Comparison With Existing Approaches

The comparison of our proposed approach with existing approaches is summarized in Table I. Our approach falls into the category of post-training augmentation (PTA), which does not require a special training process such as randomizing parameters and adding disturbances, and allows the training to be done in both simulated and real-world environments, as opposed to robust/adversarial training (RAT) methods. Additionally, RAT methods aim to find a fixed policy for all possible realizations of uncertainties, which could be infeasible when the range of uncertainties is large. Compared to existing PTA methods based on MRAC and DOB, our approach is able to deal with a broader class of uncertainties, and is validated on real hardware.

On the other hand, similar to other PTA methods, our approach needs the dynamics to be continuous and have a control-affine form, and can only effectively compensate for the matched disturbance. Dealing with the unmatched disturbances in the nonlinear setting has been a long-standing challenging problem for adaptive or DOB-based control methods, other methods, e.g., those based on robust control [33], must be considered. As a result, when the unmatched disturbance dominates the total disturbance, the performance of the proposed approach will be limited. This is demonstrated in Section IV, e.g., in the quadrotor example in the presence of wind disturbances.

IV. EXPERIMENTS

We now apply the proposed approach to two systems, namely a Pendubot and a 3-D quadrotor. In particular, for the Pendubot, experiments on real hardware are also conducted. Additional computation and simulation results for a cart-pole system are included in [32]. An overview of the systems and test settings is given in Table II. The dynamic models for these systems are included in Appendix B of [32].

TABLE II
AN OVERVIEW OF TESTING SYSTEMS AND SETTINGS

System	**State/input Dimension**	**Policy Search Methods**	**Test Environments**	**W/ Unmatched Disturbances**
Pendubot	4/1	PILCO, SAC & DR-SAC	Simulation (IV-A) & Hardware (IV-C)	Yes
Quadrotor	12/4	DDP	Simulation (IV-B)	Yes

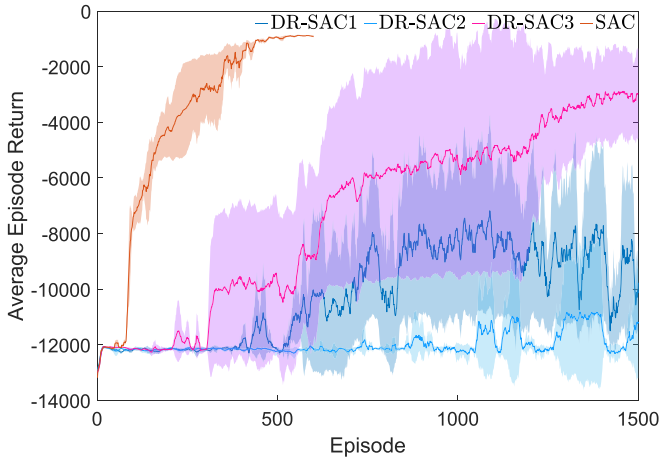

Fig. 2. Training curves for Pendubot. Shaded areas denote the variance over five trials.

A. Pendubot Swing-Up and Balance in Simulations

As depicted in Fig. 6, the Pendubot is a mechatronic system consisting of two rigid links interconnected by revolute joints with the second joint unactuated. The states of the system include the angles and angular rates of the two links, and the control input is the torque applied to Link 1. The task is to swing up the links from initial states $[q_1, q_2] = [\pi, \pi]$ to the right-up position $[q_1, q_2] = [0, 0]$ and balance them there, as illustrated in Fig. 6. The same reward function is used for training SAC and DR-SAC policies and defined by

$$r = -3(|\sin(q_1)| + |\cos(q_1) - 1| + |\sin(q_2)| + |\cos(q_2) - 1|). \tag{9}$$

The nominal RL policies were trained in simulation using soft actor-critic (SAC) [28] implemented in the MATLAB Reinforcement Learning Toolbox. For comparison, we also trained a few robust policies (termed as DR-SAC) with SAC and domain randomization [5], [6], in which three parameters, namely, the input gain (Λ), the mass of Link 1 (m_1), and the mass of Link 2 (m_2), were randomly sampled in a variety of ranges.

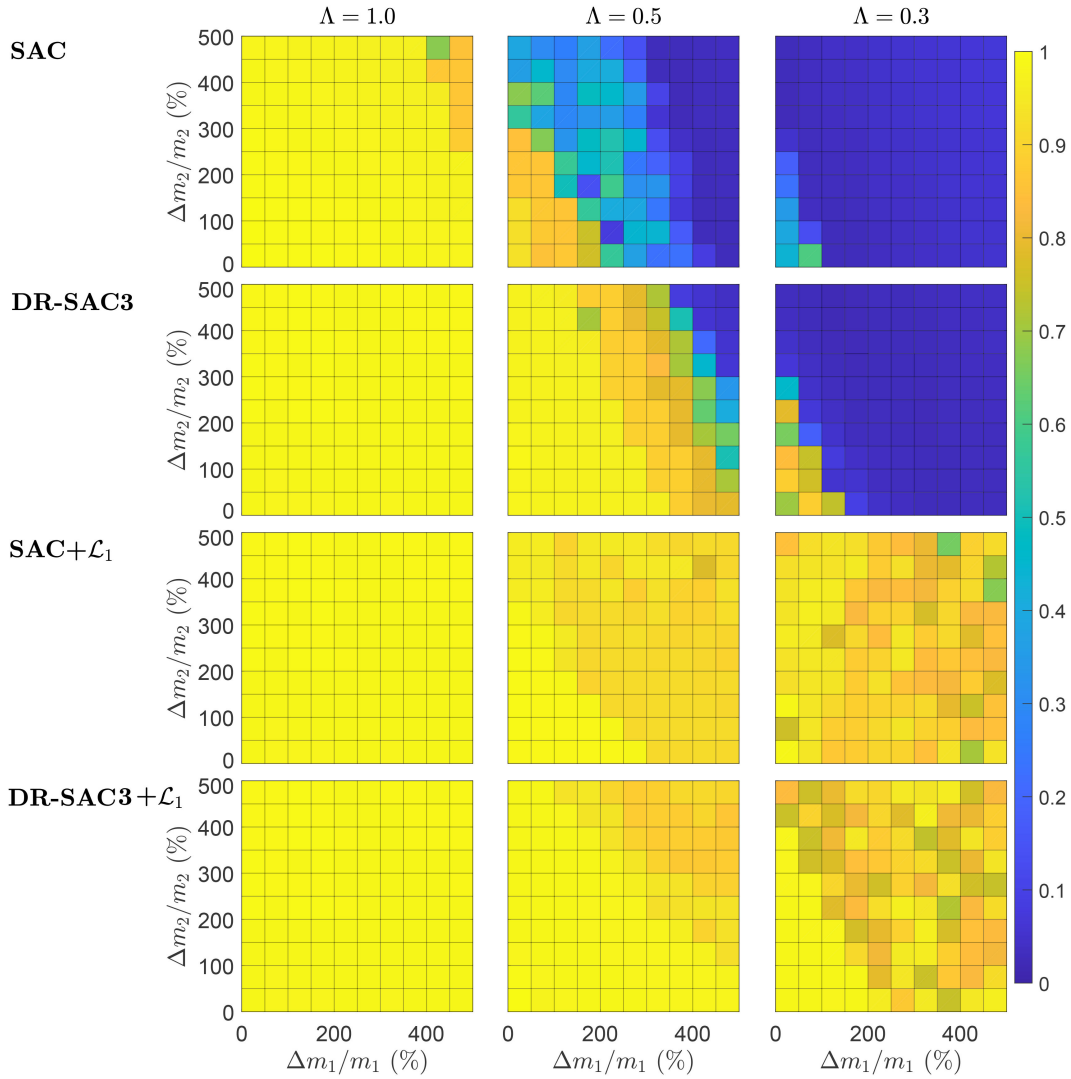

Fig. 3. Performance of SAC, DR-SAC3, SAC+$\mathcal{L}_1$ and DR-SAC3+$\mathcal{L}_1$ for Pendubot under perturbations in m_1, m_2 and Λ. Percentage change with respect to the nominal value is used to measure the perturbations in m_1 and m_2.

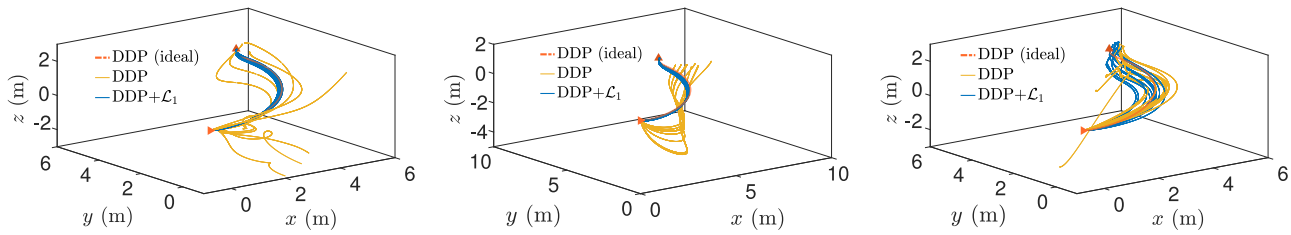

Fig. 4. Results under loss of propeller efficiency (a), perturbations in quadrotor mass and inertia (b), and wind disturbances (c). DDP (ideal) denotes the trajectory obtained by applying the policy to the nominal dynamics.

Additionally, we tried imposing different control limits (through squashing). When training the SAC and DR-SAC polices, each agent includes an actor and two critics, all three of which share the same neural network structure that has two hidden fully-connected layers with 300 and 400 neurons, respectively. The same hyper-parameters were used for training all the DR-SAC and SAC policies. We did five trials for each setting. Table III lists three of many settings that we tested for training the DR-SAC policies and the setting for training the vanilla SAC policy. Fig. 2 shows the average episode return (computed using a window of 10 episodes) during training. The solid curves correspond to the mean and the shaded region to the minimum and maximum

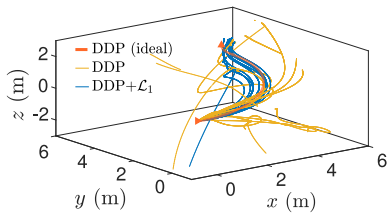

Fig. 5. Results under joint perturbations in quadrotor mass, inertia and propeller efficiencies, and wind disturbances. In each of the ten scenarios, each type of perturbation was generated in the same way as was for the results in Fig. 4(a)-4(c).

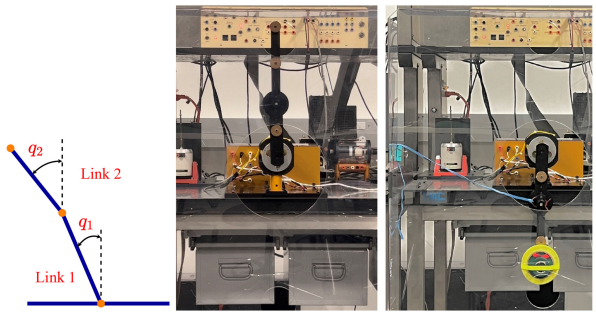

Fig. 6. Left: a Pendubot configuration. Middle: stabilization at the upright position. Right: added masses and a rubber band used to induce dynamic variations.

TABLE III
SELECTED TRAINING SETTINGS FOR PENDUBOT

Setting	Parameters	Input Limit	Policy
I	$\Lambda = 1.0$, $m_1 = 0.12$ kg, $m_2 = 0.11$ kg	4 Nm	SAC
II	$\Lambda \in [0.3, 1]$, $m_1 \in 0.12[1, 6]$ kg, $m_2 \in 0.11[1, 6]$ kg	6 Nm	DR-SAC1
III	$\Lambda \in [0.3, 1]$, $m_1 \in 0.12[1, 6]$ kg, $m_2 \in 0.11[1, 6]$ kg	9 Nm	DR-SAC2
IV	$\Lambda \in [0.5, 1]$, $m_1 \in 0.12[1, 4]$ kg, $m_2 \in 0.11[1, 4]$ kg	6 Nm	DR-SAC3

average return over the five trials. As seen in Fig. 2, it was much easier and took much less episodes to find a good SAC policy, compared to training DR-SAC policies. We were able to find a good DR-SAC policy (i.e., DR-SAC3) under Setting IV, while further increasing the range of parameter perturbations associated with Setting IV led to degraded performance of the resulting DR-SAC policies even with a larger control limit, as illustrated by the training curves for DR-SAC1 and DR-SAC2. For subsequent tests, we chose the best DR-SAC3 from all five trials and compared it with other control policies.

We tested the performance of vanilla SAC, DR-SAC, SAC with $\mathcal{L}_1$ augmentation (SAC+$\mathcal{L}_1$) and DR-SAC with $\mathcal{L}_1$ augmentation (DR-SAC+$\mathcal{L}_1$) under a wide range of perturbations in m_1, m_2, and under three input gain settings: $\Lambda = 1.0$, 0.5 and 0.3, while the latter two indicate a loss of control effectiveness by 50% and 70%, respectively. For $\mathcal{L}_1$ augmentation design, the parameters in (5), (6), and (8) were chosen to be $a = 10, T = 0.005$ second and $K = 200$, and *fixed* across all the tests. The results in terms of the normalized accumulative reward under each test scenario are shown in Fig. 3. Note that perturbation in m_2 induces unmatched uncertainties that cannot be compensated by the $\mathcal{L}_1$ control law. As one can see, the performance of vanilla SAC drops dramatically when the perturbations in m_1, m_2 and Λ increase. DR-SAC3 achieved acceptable performance under $\Lambda = 0.5$ in general, except when the perturbations in m_1 and m_2 are near the maximum, which are beyond the perturbations encountered during training of DR-SAC3. However, when the control effectiveness further decreases to 30% of its nominal value, DR-SAC3's performance degrades significantly, while only slight performance degradation is observed under SAC+$\mathcal{L}_1$ and DR-SAC3+$\mathcal{L}_1$ when the perturbations increase to the maximum. It is worth noting that SAC+$\mathcal{L}_1$ and DR-SAC3+$\mathcal{L}_1$ show comparable performance under the tested scenarios. We conjecture that in the case of larger unmatched uncertainties, DR-SAC3+$\mathcal{L}_1$ will outperform SAC+$\mathcal{L}_1$.

B. 3-D Quadrotor Navigation in Simulations

The states include quadrotor position (x, y, z) and linear velocities $(\dot{x}, \dot{y}, \dot{z})$ in an inertia frame and the roll, pitch, and yaw angles (ϕ, θ, ψ) of the quadrotor body frame with respect to the inertial frame, as well as their derivatives. Motor mixing is also included in the dynamics. The inputs are the total thrusts f_z and three moments along three axes $(\tau_\phi, \tau_\theta, \tau_\psi)$ generated by the four propellers.

The nominal value of the key parameters are set to be $[I_x, I_y, I_z] = [0.082, 0.0845, 0.1377]$ kgm^2 (moment of inertia), $m = 4.34$ kg (quadrotor mass), and $c_{pi} = 1$ $(i = 1, 2, 3, 4)$ (propeller control coefficients). The mission is to control the quadrotor to fly from the origin to the target point (4,4,2). To obtain a policy for achieving the mission, we chose to use trajectory optimization, which, together with model learning, is commonly used for model-based RL [34], [35]. We further chose to use differential dynamic programming (DDP) [29], a specific trajectory optimization method. Since our focus is not on the training but on robustifying a pre-trained policy, we use the physics-based dynamic model with the nominal parameter values as the model "learned" in the nominal environment. This model is used for computing the DDP policy, and for designing the adaptive augmentation. For computing the DDP policy, we discretized the nominal dynamics and applied the method in [29] with the cost function $J = \tilde{x}_N^\top P_N \tilde{x}_N + \sum_{i=0}^{N-1} (\tilde{x}_i^\top P \tilde{x}_i + u_i^\top Q u_i)$, where $\tilde{x}_i = x_i - x_{target}$ for $i = 1, \ldots, N$, N is the control horizon, and $P = \text{diag}(2, 2, 2, 0.1, 0.1, 0.3, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)$, $P_N = \text{diag}(10, 10, 10, 5, 5, 5, 5, 5, 5, 5, 5, 5)$ and $Q = \text{diag}(20, 4, 4, 4)$. For $\mathcal{L}_1$ augmentation design, the parameters in (5), (6), and (8) were chosen to be $a = 10, T = 0.001$ second and $K = 200$, and *fixed* across all the tests.

We tested the performance of the DDP policy with and without $\mathcal{L}_1$ augmentation under three types of dynamic perturbations. The first one is **loss of propeller efficiency**, which mimics the effect of propeller failures, and is simulated by adjusting the control coefficients c_{pi} $(i = 1, 2, 3, 4)$. Fig. 4(a) shows the resulting trajectories under ten scenarios, in each of which the control coefficients of two propellers were randomly selected to be in [0.5,1]. One can see that $\mathcal{L}_1$ augmentation significantly improved the robustness of the DDP policy, leading to consistent trajectories that are close to the ideal trajectory obtained by applying the policy to the nominal dynamics. The second type

TABLE IV
TEST RESULTS UNDER DIFFERENT SCENARIOS

Scenario \ Policy	PILCO	PILCO+$\mathcal{L}_1$	SAC	SAC+$\mathcal{L}_1$	DR-SAC	DR-SAC+$\mathcal{L}_1$
I: Nominal	✔	✔	✔	✔	✔	✔
II: $\Lambda = 0.5$	✗	✔	✗	✔	✔	✔
III: Added Masses of $270g$ (100% of m_2)	✔	✔	✔	✔	✔	✔
IV: $\Lambda = 0.6$ plus Added Mass of $90g$ (33% of m_2)	✗	✔	✗	✔	✔	✔
V: $\Lambda = 0.5$ plus Added Masses of $450g$ (167% of m_2)	✗	✔	✗	✗	✗	✔
VI: Added disturbances with a rubber band	✗	✔	✗	✔	✔	✔

of dynamic perturbations are the **mass and inertia change**, e.g., to mimic the effect of carrying different packages for a delivery drone. Fig. 4(b) shows the results under ten scenarios with randomly increased mass and inertia through a scale of [2,5]. Once again, $\mathcal{L}_1$ augmentation significantly improved the policy robustness, leading to close-to-ideal trajectories. The third type of dynamic variations is related to **wind disturbances** in the horizontal plane, which causes disturbance forces in the x and y directions. In each of the ten scenarios, the forces were simulated by stochastic variables with the mean values randomly sampled from [10,25]. The results are depicted in Fig. 4(c). $\mathcal{L}_1$ augmentation improved the robustness, but was not able to yield close-to-ideal performance. This is mainly because the wind disturbances induce unmatched disturbances ($\hat{\sigma}_{um}(t)$ in (5) and (6)), which are not compensated for in the control law (8). Finally, Fig. 5 illustrates the simulation results under **joint perturbations in quadrotor mass, inertia and propeller efficiency and wind disturbances**.

C. Pendubot Swing-Up and Balance on Real Hardware

We further tested the performance of those policies used in Section IV-A on the hardware setup depicted in Fig. 6. In addition to SAC and DR-SAC, we trained another policy using PILCO with the same reward function defined by (9). The ways to introduce dynamic variations include changing the input gain Λ, adding masses to Link 2, adding disturbance forces using a rubber band and different combinations of these three ways. For $\mathcal{L}_1$ augmentation design, the parameters in (5), (6), and (8) were chosen to be $a = 150$, $T = 0.005$ s and $K = 150$ for most of the policies in most of the test scenarios. For DR-SAC in Test I, $K = 100$ (corresponding to a lower bandwidth for the low-pass filter) was used to avoid large vibrations at the upright position, due to the fact that DR-SAC has a relatively high gain to attenuate the effect of dynamic variations. The test scenarios and results are summarized in Table IV, where ✔ (✗) indicates a success (failure) in achieving the mission. A video of the experiments is available at https://youtu.be/xZBcsNMYK3Y.

As one can see, in the nominal case (i.e., without intentionally introduced dynamic variations), all the policies with and without $\mathcal{L}_1$ augmentation succeeded in achieving the mission. This, to a certain extent, indicates that the $\mathcal{L}_1$ augmentation does not adversely affect the performance of RL policies in the presence of no or minimal dynamic variations. Additionally, $\mathcal{L}_1$ augmentation significantly improves the robustness of PILCO and vanilla SAC, enabling them to succeed under all the tested scenarios except Scenario V for SAC, due to the extreme dynamic variations induced by the the largest perturbations in input gain and added masses. DR-SAC displayed much more robustness compared to vanilla SAC as expected, and only failed under Scenario V. It's worth noting that $\mathcal{L}_1$ augmentation also further enhanced the robustness to DR-SAC and made it succeed under Scenario V. In Scenario VI, a rubber band was attached to the joint connecting the two links to exert a disturbance force. The disturbance force applied by the rubber band changed quite rapidly and peaked when Link 1 reached the upright position. This caused great challenges for the RL policies, as evidenced by the struggling of PILCO and SAC in the video, since, by training, these policies are not expected to produce large control inputs near the upright position. Nevertheless, with the help of $\mathcal{L}_1$ compensation, PILCO and SAC were able to deal with this challenging scenario.

V. CONCLUSION

This paper presents an add-on scheme to improve the robustness of a reinforcement learning (RL) policy for controlling systems with continuous state and action spaces, by augmenting it with an $\mathcal{L}_1$ adaptive controller ($\mathcal{L}_1$AC) that can quickly estimate and actively compensate for potential dynamic variations during execution of this policy. Our approach is easy to implement and allows for the policy to be trained or computed using almost any RL method (model-free or model-based), either in a simulator or in the real world, as long as a control-affine model to describe the dynamics of the nominal environment is available for the $\mathcal{L}_1$AC design. Experiments on different systems in both simulations and on real hardware demonstrate the general applicability of the proposed approach and its capability in improving the robustness of RL policies including those trained robustly, e.g., using domain/dynamics randomization (DR). Future work includes incorporating mechanisms, e.g., based on robust control [31], [33], to mitigate the effect of unmatched disturbance, and model learning to safely and robustly learn the unknown dynamics.

The proposed approach and existing robust RL methods e.g., based on DR, do not necessarily replace each other. Instead, they can complement each other, as demonstrated by the experimental results in Section IV-C. As mentioned before, existing robust RL methods aim to find a *fixed* policy for all possible realizations of uncertainties, which could be infeasible when the range of uncertainties is large. On the other hand, the proposed adaptive augmentation approach can deal with significant amount

of matched uncertainties by using additional control effort to actively compensate for those, but cannot handle unmatched uncertainties in its current form. For systems subject to both matched and unmatched disturbances, a compelling solution will be to combine the strength of both by (1) (partially) ignoring matched disturbances in training a policy using existing robust RL methods to reduce conservativeness, and (2) augmenting the trained policy with the proposed $\mathcal{L}_1$ scheme during execution of this policy to compensate for matched disturbances.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[2] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Proc. Conf. Robot Learn.*, 2018, pp. 133–145.

[3] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872.

[4] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

[5] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3803–3810.

[6] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Trans. Robot.*, vol. 36, no. 1, pp. 1–14, Feb. 2020.

[7] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2817–2826.

[8] M. A. Abdullah *et al.*, "Wasserstein robust reinforcement learning," 2019, *arXiv:1907.13196*.

[9] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8093–8104.

[10] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[11] J. W. Kim, H. Shim, and I. Yang, "On improving the robustness of reinforcement learning-based controllers using disturbance observer," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 847–852.

[12] A. Guha and A. Annaswamy, "MRAC-RL: A framework for on-line policy adaptation under parametric model uncertainty," 2020, *arXiv:2011.10562*.

[13] N. Hovakimyan and C. Cao, $\mathcal{L}_1$ *Adaptive Control Theory: Guaranteed Robustness With Fast Adaptation*. Philadelphia, PA, USA: SIAM, 2010.

[14] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, "$\mathcal{L}_1$-adaptive MPPI architecture for robust and agile control of multirotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7661–7666.

[15] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. A. Theodorou, "$\mathcal{L}_1$-$\mathcal{GP}$: $\mathcal{L}_1$ adaptive control with Bayesian learning," in *Proc. 2nd Annu. Conf. Learn. Dyn. Control*, 2020, pp. 826–837.

[16] A. Lakshmanan, A. Gahlawat, and N. Hovakimyan, "Safe feedback motion planning: A contraction theory and $\mathcal{L}_1$-adaptive control based approach," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 1578–1583.

[17] A. Gahlawat *et al.*, "Contraction $\mathcal{L}_1$ adaptive control using Gaussian processes," in *Proc. 3rd Conf. Learn. Dyn. Control*, 2021, pp. 1027–1040.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 112 6–11.

[19] A. Nagabandi *et al.*, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *Proc. Int. Conf. Learn. Representations* , 2019.

[20] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based RL," in *Proc. Int. Conf. Learn. Representations*, 2019.

[21] S. Sæmundsson, K. Hofmann, and M. P. Deisenroth, "Meta reinforcement learning with latent variable Gaussian processes," in *Proc. Conf. Uncertainty Artif. Intell.*, 2018.

[22] M. Xu, W. Ding, J. Zhu, Z. LIU, B. Chen, and D. Zhao, "Task-agnostic online reinforcement learning with an infinite mixture of Gaussian processes," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6429–6440.

[23] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Mineola, NY, USA: Dover, 2012.

[24] W.-H. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-observer-based control and related methods—An overview," *IEEE Trans. Ind. Electron.*, vol. 63, no. 2, pp. 1083–1095, Feb. 2016.

[25] R. Takano and M. Yamakita, "Robust control barrier function for systems affected by a class of mismatched disturbances," *SICE J. Control, Meas., Syst. Integration*, vol. 13, no. 4, pp. 165–172, 2020.

[26] M. J. Khojasteh, V. Dhiman, M. Franceschetti, and N. Atanasov, "Probabilistic safety constraints for learned high relative degree system dynamics," in *Proc. Annu. Conf. Learn. Dyn. Control*, 2020, pp. 781–792.

[27] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 465–472.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. ICML*, 2018, pp. 1861–1870.

[29] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4906–4913.

[30] P. Zhao, Y. Mao, C. Tao, N. Hovakimyan, and X. Wang, "Adaptive robust quadratic programs using control Lyapunov and barrier functions," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 3353–3358.

[31] P. Zhao, S. Snyder, N. Hovakimyana, and C. Cao, "Robust adaptive control of linear parameter-varying systems with unmatched uncertainties," 2021, *arXiv:2010.04600*.

[32] Y. Cheng, P. Zhao, F. Wang, J. D. Block, and N. Hovakimyan, "Improving the robustness of reinforcement learning policies with $\mathcal{L}_1$ adaptive control," 2021, *arXiv:2106.02249*.

[33] P. Zhao, A. Lakshmanan, K. Ackerman, A. Gahlawat, M. Pavone, and N. Hovakimyan, "Tube-certified trajectory tracking for nonlinear systems with robust control contraction metrics," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5528–5535, Apr. 2022.

[34] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 829–837.

[35] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1907–1915.